Troubleshooting Windows Services

This page describes how to troubleshoot Windows Services.

On this page:

Missing Libraries for a Windows Service

One failure that commonly occurs when starting a Windows service is caused by missing DLLs, which usually results in an error window stating a particular DLL cannot be found. Fixing this problem can often be a trial-and-error process because the DLL mentioned in the error may depend on other DLLs that are also missing. It is important to understand that a Windows service is launched by the operating system and can be configured to execute as a different user, which means the service's environment (most importantly its `PATH`) may not match yours and therefore extra steps are necessary to ensure that the service can locate its required DLLs.

> The command-line utility `dumpbin` can be used to discover the dependencies of an executable or DLL.

The simplest approach is to copy all of the necessary DLLs to the directory containing the service executable. If this solution is undesirable, another option is to modify the system `PATH` to include the directory or directories containing the required DLLs. (Note that modifying the system `PATH` requires restarting the system.) Finally, you can copy the necessary DLLs to `\WINDOWS\system32`, although we do not recommend this approach.

> Copying DLLs to `\WINDOWS\system32` often results in subtle problems later when trying to develop using newer versions of the DLLs. Inevitably you will forget about the DLLs in `\WINDOWS\system32` and struggle to determine why your application is misbehaving or failing to start.

Assuming that DLL issues are resolved, a Windows service can fail to start for a number of other reasons, including

- invalid command-line arguments or configuration properties
- inability to access necessary resources such as file systems and databases, because either the resources do not exist or the service does not have sufficient access rights to them
- networking issues, such as attempting to open a port that is already in use, or DNS lookup failures

Failures encountered by the Ice run time prior to initialization of the communicator are reported to the Windows event log if no other logger implementation is defined, so that should be the first place you look. Typically you will find an entry in the `System` event log resembling the following message:

```
The IcePatch2 service terminated with service-specific error 1.
```

Error code `1` corresponds to `EXIT_FAILURE`, the value used by the `Service` class to indicate a failure during startup. Additional diagnostic messages may be available in the `Application` event log. See `Ice::Service` Logging Considerations for more information on configuring a logger for a Windows service.

As we mentioned earlier, insufficient access rights can also prevent a Windows service from starting successfully. By default, a Windows service is configured to run under a local system account, in which case the service may not be able to access resources owned by other users. It may be necessary for you to configure a service to run under a different account, which you can do using the Services control panel. You should also review the access rights of files and directories required by the service.

Windows Firewall Interference

Your choice of user account determines whether you receive any notification when the Windows Firewall blocks the ports that are used by your service. For example, if you use `Local Service` as we recommended, you will not see any Windows Security Alert dialog (see this Microsoft article for details).

If you are not prompted to unblock your service, you will need to manually add an exception in Windows Firewall. For example, follow the steps below to unblock the ports of a Glacier2 router service:

1. Open the Windows Firewall Settings panel and navigate to the Exceptions panel.
2. Select "Add program..."
3. Select "Browse," navigate to the Glacier2 router executable, and click "OK."

Note that adding an exception for a program unblocks all ports on which the program listens. Review the endpoint configurations of your services carefully to ensure that no unnecessary ports are opened.

For services listening on one or a few fixed ports, you could also create port exceptions in your Windows Firewall. Refer to the Windows Firewall documentation for details.

IceGrid Node Performance Monitoring Issues

The IceGrid node uses Windows' `Perflib` facility to obtain statistics about the CPU utilization of its host for load balancing purposes. On Vista-derived operating systems, the IceGrid node may log the following warning message:

```
warning: Unable to lookup the performance counter name
```

This message is an indication that the node does not have sufficient privileges to access a key in the Windows registry:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib
```

As part of its installation procedure, the `iceserviceinstall` utility modifies the permissions of this registry key to grant read access to the node's designated user account. If you are trying to change the node's user account, we recommend using the `iceserviceinstall` utility to uninstall and reinstall the node. If you wish to modify the permissions of this registry key manually, follow these steps:

1. Start `regedit` and navigate to the `Perflib` key.
2. Right click on `Perflib` and select `Permissions`.
3. If the desired user account is not already present, click `Add` to add the user account. Enter `LOCAL SERVICE` if you wish to run the node in the Local Service account, otherwise enter the name of the user account. Press `OK`.
4. Check the `Read` box in the `Allow` column to grant read access to the registry key and press `OK` to apply the changes.

Another way to grant the node's user account with the necessary access rights is to add it to the `Performance Monitor Users` group.

See Also

- Load Balancing
- Installing a Windows Service
- Manually Installing a Service